RL-TR-95-50
Final Technical Report
March 1995

RADAR SIGNAL PROCESSING RULEBASE PARTITIONING

Pragati Synergetic Research, Inc.

Mala Mehrotra

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-95-50 has been reviewed and is approved for publication.

APPROVED: William J. Baldygo, Jr.

WILLIAM J. BALDYGO, Jr.
Project Engineer

FOR THE COMMANDER: Donald W. Hanson

DONALD W. HANSON
Director of Surveillance & Photonics

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (OCSS) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1995	Final Mar 94 - Jan 95

4. TITLE AND SUBTITLE
RADAR SIGNAL PROCESSING RULEBASE PARTITIONING

5. FUNDING NUMBERS
C - F30602-94-C-0069
PE - 61102F
PR - 2304
TA - E8
WU - 07

6. AUTHOR(S)
Mala Mehrotra

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Pragati Synergetic Research, Inc.
145 Stonelake Court
Yorktown VA 23693

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (OCSS)
26 Electronic Parkway
Griffiss AFB NY 13441-4514

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-95-50

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: William J. Baldygo, Jr./OCSS (315) 330-4049

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This technical report addresses issues involved in the integration, enhancement and porting of two large knowledge-based systems, (1) Expert System Constant False Alarm Rate (ES-CFAR) and (2) The Integrated Multi-Domain Radar Demonstration (IMRD), to more current software/hardware platforms using sound software engineering principles.

14. SUBJECT TERMS
Artificial Intelligence, Expert Systems, Radar Signal Processing

15. NUMBER OF PAGES
40

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	U/L

1 Introduction

This report addresses issues involved in the integration, enhancement and porting of two large knowledge-based systems - Constant False Alarm Rate Expert System (CFAR-ES) and Integrated Multi-Domain Radar Demonstration Expert System (IMRD-ES) - to more current software/hardware platforms, using sound software engineering principles. Our objective is to formulate the set of desirable requirements for the new expert system shell to be used for implementing the integrated code. In order to achieve these goals, a careful analysis of the two systems had to be undertaken. Multi-viewpoint methodology was used for this purpose as it is able to abstract, structure and cluster a large knowledge-based system in a manner that facilitates its understanding, manipulation, testing and utilization. The multi-view point methodology is founded on the principle that *no single structuring principle or abstraction hierarchy is sufficient to understand complex knowledge-bases.* The presentation of a knowledge-based system from several different viewpoints gives very valuable leverage for various life-cycle activities by making the knowledge-based system more comprehensible and tractable. Much of any software life-cycle activities depend upon the right modularization for more efficient analysis.

Initial study of the two systems revealed that the procedural aspects in both the systems are intermingled very tightly with the knowledge-based aspects of the problem. In the IMRD-ES system a lot of procedural calls to Pascal software exists from Prolog. In the CFAR-ES code the rulebase portion of Gensym's G2 code are meshed in tightly with procedural code written

in C. Neither Gensym's G2 nor Prolog provides any structuring mechanisms for code, thus making it very difficult for long-term maintenance. We were able to analyze the AI aspects of the IMRD-ES code. However, CFAR-ES code could not be analyzed as the rules could not be separately extracted and analyzed. In any case, there were only a few embedded rules in the CFAR-ES code.

We have used the results of our analysis of IMRD-ES along with documentation for CFAR-ES to identify functional commonalities between the two systems and to suggest possible ways of integrating them. Our analysis also suggests that a more structured environment needs to be chosen for the integration and porting of the two systems. If AI aspects such as fuzzy logic, uncertainty, non-monotonic reasoning etc. has to be incorporated, at present or in future, then an expert system shell which incorporates these features needs to be chosen. There is ample evidence that radar returns are often contaminated enough by weather, clutter and jammers that a backward chaining system with fuzzy logic reasoning will be desirable.

In this report we will first outline Rome Laboratories' two software systems. Next we will outline our software analysis tool called MVP-CA (Multi-View Point - Clustering Analysis). We will then discuss the software engineering aspects of these systems as revealed by the MVP-CA tool, followed by suggestions for future enhancements in these codes.

2 Structure of IMRD-ES and CFAR-ES

Rome Laboratory utilizes two independently developed software systems in their radar target detection environment. The first is the Constant False Alarm Rate Expert System (CFAR-ES) that aids the radar surveillance system by choosing the appropriate CFAR algorithm for target detection with minimal false detections. The second is the Integrated Multi-Domain Radar Demonstration Expert System, which automatically controls various modes of operation and parameters for the multi-function C-Band phased array radar system.

CFAR-ES adaptively selects the appropriate CFAR algorithm for accurate target detections. The selection process is dependent on the various characteristics of the observed radar data. Preprocessing routines classify the radar data into various homogeneous clutter regions. AI rules in CFAR-ES integrates this information with data from geographical maps, tracker information, as well as inputs from the user, to determine the appropriate CFAR algorithm to be executed for detection. It also determines the appropriate CFAR parameter values for the CFAR processor. CFAR-ES is currently written in Gensym's G2 shell with computations handled by remote C procedure calls. Very few rules are written in G2 and almost all of them are inside C procedures.

IMRD-ES identifies interference in radar returns and recommends the appropriate responses for the next scan. The surveillance region is partitioned into threat zones prioritized by the importance of maintaining detection within them. The beam dwell time is adaptively apportioned to different

beams, subject to a desired scan time constraint. The objective is to identify targets in the presence of clutter and jammer interference. Inputs to the IMRD-ES is the Beam Scan Data processed by the ST-100 array processor. The knowledge-based adaptive controller in IMRD-ES (expressed as 337 Prolog predicates) analyzes the data for sources of interference, determines the control changes to be made for the next scan to maximize the probability of detection in conjunction with the prioritized regions of coverage. The variable dwell time per beam is divided into three processing intervals: a passive listening interval, CPI1, for detecting jammers and radio frequency interference sources; an active environmental assessment interval, CPI2, for detecting ground and weather clutter; and an adaptive target detection waveform, CPI3, to apply appropriate electronic counter measures, as determined by the expert system. A single fixed-parameter CFAR algorithm is used for target detection in the present system. The expert system resides on a VAX currently and has been tested to perform with both real and simulated data.

3 MVP-CA

3.1 Motivation for Clustering Knowledge-Based Systems Software

Knowledge-based systems owe their appeal to the promise of utilizing expertise in the domain knowledge for the solution of difficult, poorly-understood, ill-structured problems. However, they must be subjected to rigorous verification and validation (V&V) analyses before they can be accepted into

real-world critical applications. Unfortunately, expert systems do not lend themselves to the traditional V&V techniques for highly reliable software. There is a need to formulate an acceptable set of V&V techniques which can assure their quality. Better knowledge-acquisition techniques as well as better management, understanding and enhancement of the knowledge base is critical to the success of such V&V activities.

The difficulty in the V&V of large knowledge-based systems arises due to a number of reasons. Firstly, rapid prototyping and iterative development form key features of any expert system development activity. This has led to the development of ad-hoc techniques for expert system design without any software engineering guidelines. Moreover, due to the data-driven nature of expert systems, as the number of rules of an expert system increase, the number of possible interactions between the rules increases exponentially. The complexity of each pattern in a rule compounds the problem of V&V even further. As a result, large expert systems tend to be incomprehensible, difficult to debug or modify, and almost impossible to verify or validate.

Compounding the problem further is the fact that most expert systems are built without much regard to defining the requirements or specifications. As any software, conventional or knowledge-based, becomes more complex, common errors are bound to occur through misunderstanding of specifications and requirements. Large computer systems are typically built by multiple teams who may have different, yet legitimate perspectives of the system design. Often, conflicting opinions on implementational aspects of the system will need to be resolved. Ambiguities and interpretational problems will always be inherent to large systems software development. Human limita-

tions in understanding and managing software complexity will give rise to deficiencies in software. In addition, since knowledge continues to remain in an evolving state for large systems, with new knowledge augmenting the software all the time, it is essential to provide a mechanism to check the conceptual aspects of the system from time to time. Studies have shown that the majority of time spent in maintaining existing systems lies in understanding the software and how it relates to the application domain. Too many systems were built with poor software engineering practices or have been changed so many times that gaining this understanding is very difficult and is one of the most common means by which faults are introduced into the system.

Therefore, it is our belief that even if a software life cycle stresses specifications and requirements upfront, that will not be enough to guarantee the right product for complicated systems. There are bound to be ambiguities and interpretational problems. What is needed is a complementary tool that is capable of exposing such ambiguities and misinterpretations so that corrective action can be taken before it is too late in the software life cycle. Having a semi-automated means of capturing and structuring the meta-knowledge in a rulebase and cross-checking it with the specifications and requirements at various stages of the software life cycle could certainly help in this effort.

Conventional software yields more easily to verification efforts because control is explicitly represented as procedures which can be structured to encapsulate run-time abstractions. Modules can be designed in conventional software, each consisting of a manageable unit with a well-defined interface. Furthermore, procedures can be grouped into packages or objects which share an internal data structure. These units can then be subjected

to unit/integration testing techniques. Isolating special-purpose variables in a knowledge-based system into smaller subknowledge bases can lead to a reduction in the number of test cases to be designed. We have demonstrated this aspect through our experimental results which has been discussed in [6].

Due to the declarative style of programming in knowledge-based systems, the generation of clusters to capture significant concepts in the domain seems more feasible than it would be for procedural software. By using knowledge-based programming techniques one is much closer to the domain knowledge of the problem than with procedural languages. The control aspects of the problem are abstracted away into the inference engine (or alternatively, the control rules are explicitly declared). The existence of a model of the domain would benefit the analysis of other knowledge-based systems within that domain by providing seeds for cluster formation. Moreover, transfer of expertise from one problem domain to another related domain can be facilitated through the factoring of common aspects across domains. Hence software reuse can be exploited through multiple structuring of a knowledge-based system. It is our contention that such activities will require that large complex knowledge-based systems to be viewed from several different, possibly orthogonal viewpoints. In addition, the use of a domain model to assist in the development of new knowledge-based systems is a promising research direction.

Existing research indicates that misunderstandings of the domain are a primary cause of systems failures [1, 4, 13]. Often small oversights or misunderstood interactions between sources of expertise lead to catastrophic failures. Even though language support for systems structuring has long

been recognized as a key aspect of modern software and knowledge engineering, it is our contention that *no single structuring can simultaneously capture all the important concepts in complex knowledge-based systems.* We believe that techniques, methodologies and supporting tools are needed to manage a complex system from multiple viewpoints and that the discovery of subtle interrelating concepts is critical for assuring the reliability of these systems.

Multi-Viewpoint Clustering Analysis (MVP-CA) is a feasible and effective technique for structuring a rulebase in various meaningful ways so as to capture its explicit as well as implicit knowledge. Preliminary results obtained by clustering several knowledge-bases have shown that *no single structuring principle or abstraction hierarchy is sufficient to understand complex knowledge bases* [9, 10, 11, 12]. Multi-ViewPoint-Clustering Analysis (MVP-CA) methodology provides multiple views of the same expert system. Significant structures within the rulebase are discovered by structuring the system both hierarchically (from detail to abstract) and orthogonally (from different perspectives).

Our approach utilizes clustering analysis techniques to group rules which share significant common properties and to identify the concepts underlying in these groups. Cluster analysis is a kind of unsupervised learning in which (a potentially large volume of) information is grouped into a (usually much smaller) set of clusters. If a simple description of the cluster is possible, then this description emphasizes critical features common to the cluster elements while suppressing irrelevant details. Thus, clustering has the potential to abstract from a large body of data, a set of underlying principles or concepts which organizes that data into meaningful classes. This

knowledge acquisition process involves "mining" the rulebase for interesting concepts shared among the rules. Our research efforts address the feasibility of providing automated support in the identification of such meaningful rule-groups in knowledge-based systems software, so as to reflect the underlying subdomains of the problem. Partitioning rule-based systems into a number of meaningful units can enhance the comprehensibility, maintainability, testability, and reliability of expert systems software significantly.

The extraction of implicit, previously unknown, yet potentially useful information from the rulebase can have considerable impact on various stages of the life cycle of knowledge-based systems software. It can expose various design pitfalls during construction of the rulebase and the functional limitations of the software during its operation, as well as the subtle inter-relationships between subgroups of rules could prove very valuable in the testing and maintenance of the system.

3.2 Overview of MVP-CA

The MVP-CA tool is divided into two phases: the *Cluster Generation Phase* and the *Cluster Analysis Phase*. In the *Cluster Generation Phase* the focus is on generating meaningful clusters through statistical and semantics-based measures. In the *Cluster Analysis Phase* the focus is on performing a statistical and functional analysis of the output generated from the previous phase. Results of this analysis are fed back into the clustering tool as better constraints on the parameters for grouping, to improve the quality of subsequent clusterings. A functional analysis of the clusters captures the key concepts

that underlie the generated clusters. *Concepts* are meaningful patterns in the rulebase along with their associated attributes or values. A set of key concepts constitutes a single *viewpoint.* Multiple *clusterings* present multiple viewpoints on the rulebase.

A two-step procedure is utilized for extracting multiple viewpoints of a rulebase. First, the best possible cluster is formed using various measures, such as distance metric, dispersion, cohesion and coupling. When group cohesiveness is plotted against number of groups, plateau regions are generated signifying stable values for cohesiveness in certain ranges of number of groups. These regions represent optimal partitionings for a particular level of conceptual abstraction. Insight into concepts dominating the various clusters can be obtained through an examination of the groups at select points on the plateau regions. A hierarchical view of the rulebase can then be generated by repeating the above procedure for different plateau regions on the cohesiveness plots.

Next, with this "best" cluster, a concept focus list is formed, to either sharpen a current viewpoint or expose an alternate viewpoint. The concept focus list is generated from dispersion statistics of patterns. Dispersion is based on shared concepts, i.e., how much a single concept is dispersed among the clusters. Low dispersion concepts are likely to represent concepts which characterize the clusters they are in. In fact, high dispersion noise concepts may interfere with the generation of highly cohesive clusters. Removing these concepts before clustering can help define the clusters more distinctly - a process which we call "sharpening." However, high dispersion concepts may represent legitimate alternate structurings of the knowledge base as well. By

selectively removing the low dispersion concepts, it is possible to reveal subtle alternate viewpoints - a concept we have termed multiview point clustering analysis [12]. Alternate viewpoints are also obtained by applying different distance metrics on the knowledge-based systems.

In the Cluster Generation Phase, the rulebase, together with the concept focus list, feeds into the frontend interpreter. The clustering algorithm starts with all rules in their own clusters. At each step of the algorithm, the two groups which are the most similar are merged together to form a new group. This pattern of mergings forms a hierarchical cluster from the single-member rule clusters to a cluster containing all the rules. In order to aid in the analysis of this hierarchy and to highlight high and low dispersion concepts, various metrics have been defined and are measured during the Cluster Generation Phase.

The following measures control and quantify the quality of clusters generated.

- distance metric
- coupling measure
- cohesiveness measure
- dispersion measure

3.2.1 Distance Metric

Distance Metric measures the relatedness of two rules in a rulebase by capturing different types of information for different classes of expert systems.

The nature of a knowledge-based system plays an important role in defining the relatedness of two rules.

The nature of the domain knowledge enforces a certain programming methodology on the developer of a rulebase [2]. Classification systems, have a hierarchical structure which yields easily to a data-flow grouping. The fundamental characteristic of such systems is that the flow of data takes place from the consequent of one rule to antecedent of other rules. For this type of rulebase, it is appropriate to use a distance metric that captures information from only right-hand side of one rule and left-hand side of the other. A monitoring system issues different commands depending on the status of different components of the system being monitored. In such systems, the antecedents of the rules usually search for special values of sensors in the component system, and the consequents assert actions to be taken when different components fail. The bulk of domain information required for grouping is usually present in the antecedents of rules in a monitoring system. This gives rise to the antecedent distance metric that captures information only from the antecedents. Alternatively, grouping the rulebase on different component failures asserted or remedial actions advocated by the consequents, gives rise to the consequent metric. The kind of distance metric to be used is a function both of the nature of the task performed by the rulebase (classification, diagnosis, control) as well as the nature of the analysis required by the user (restructuring, testing, comprehension, reuse, etc.).

We have previously shown that meaningful clusters can be formed using the number of common patterns between two rules as a measure of similarity [5, 7, 8]. Each pattern can be a word, string or number present in the

rules. The distance between two rules, which could alternatively be looked upon as the similarity between two rules, r_1 and r_2, is defined as

$$d(r_1, r_2) = \frac{Total\ no.\ of\ patterns\ in\ r_1\ and\ r_2}{no.\ of\ \text{“}common\text{”}\ patterns\ in\ r_1\ and\ r_2}$$

where different definitions of "common" give rise to different distance metrics. When there are no common patterns between r_1 and r_2, $d(r_1, r_2)$ is replaced by the maximum number of patterns allowed.

3.2.2 Coupling Measure

Our coupling measure is a measure of the average inter-group distance between two clusters. This is the factor that currently controls the choice of clusters to be merged at the next stage. This measure is based upon the distance metric used, since the distance between two groups is dependent on the distance between rules, which in turn is dictated by the metric chosen for clustering the rulebase.

The clustering algorithm starts with each rule in its own group. Groups are then merged based on the minimum inter-group distance. We define inter-group distance, $D(G_i, G_j)$ as follows:

$$D(G_i, G_j) = \sum_{r_k \epsilon G_i} \sum_{\substack{r_l \epsilon G_j \\ r_k \neq r_l}} \frac{d(r_k, r_l)}{n_i * n_j}$$

where n_i and n_j are the number of rules in groups G_i and G_j, respectively.

Using this definition of inter-group distance, we form an automatic clustering algorithm as follows. In this algorithm, the user provides the total number of groups, M, to be formed, which serves as a stopping criterion.

Even though only the total number of groups need to be specified for the clustering process, we have built zoom-in capabilities now that provide statistics when a range for the maximum number of groups to minimum number of groups is specified.

A high level view of the algorithm is given below:

Initialize each rule into its own group

While ($number\ of\ groups\ >\ M$)

Find groups G_i and G_j with minimum inter-group distance $D(G_i, G_j)$

Merge groups G_i and G_j

The average inter-group distance for a clustering C with m groups, is defined as

$$D_{aver}(C) = \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} \frac{D(G_i, G_j)}{((m * (m-1)) / 2)}$$

since there are there are *(m*(m-1))/2* distinct pairs of groups in such a clustering.

3.2.3 Cohesiveness Measure

For a given clustering, C, cohesiveness measure is an index of the similarity of rules belonging to the same group. Cohesiveness of a rule r_k with respect to the group G_i that it belongs to is the average number of concepts it shares

with the other rule members in the group G_i.

$$coh_{G_i}(r_k) = \sum_{\substack{(r_l \epsilon G_i) \\ (r_k \neq r_l)}} \frac{| 2 * common\_concepts(r_k, r_l) |}{| concepts(r_k) | + | concepts(r_l) |}$$

Cohesiveness of a group G_i is the average number of shared concepts over all rule pairs in the group,

$$coh(G_i) = \sum_{r_k \epsilon G_i} \frac{coh_{G_i}(r_k)}{n_i * (n_i - 1)},$$

where n_i is the number of rules in group G_i. Overall cohesiveness of a clustering is the cohesiveness for each group averaged over all groups for a given clustering,

$$overall\_coh(C) = \frac{\sum_{i=1}^{n_C} coh(G_i)}{n_C}$$

where n_C is the number of groups in the clustering C.

Note that cohesiveness is independent of the metric used to cluster the rulebase. The cohesion statistics feeds into a graphical tool which displays the overall cohesion of clustering at each merge point. In the preliminary stages of exploration of a knowledge-based system, cohesion plots give valuable insight into the range for optimal partitioning regions, G_{max} to G_{min}, to be examined. That is, it provides an intuitive feel for the optimal clusterings of the system where meaningful partitions exist. Removal of noise patterns from the rulebase helps sharpen the drop points of the plateaus on the cohesion plots. Distinct plateaus at different levels suggest different abstraction levels for a given viewpoint. Plateau towards the smaller number of groups suggest abstractions at higher conceptual levels.

3.2.4 Dispersion Measure

The concept focus list which is input to the formatted rulebase helps guide the clustering process by filtering out select patterns from the rulebase. This list is formed from the dispersion statistics. Dispersion measures the degree to which a single pattern is dispersed among the clusters. It is defined in the following manner:

For a single group G_i, dispersion of a pattern, p, is:

$$disp_{G_i}(p) = \begin{cases} 1 & \text{if } p \in G_i \\ 0 & \text{otherwise} \end{cases}$$

The overall dispersion of p is therefore

$$disp(p) = \sum_{i=1}^{n_C} disp_{G_i}(p)$$

where n_C is the number of groups in a clustering, C.

At any point in the clustering process, the dispersion for all the patterns can be obtained. For each pattern, dispersion gives the number of its group affiliations. As the mergings progress, a group can become stable with respect to a certain pattern or combination of patterns. When this happens the pattern or pattern combination is flagged as having become *stable*. This means that the flagged pattern or pattern combination does not occur anywhere else in the rulebase. Identification of such phenomena is very valuable in reducing the number of test cases that must be designed as this type of partitioning isolates concepts that belong together. Hence, a group-based testing strategy can be applied.

It is apparent that complete semantic information cannot be extracted from a syntactic analysis of the knowledge-based system. Thus, a fully automatic tool based on just syntactic analysis is limited by the syntactic structure of the knowledge base. In fact, one of the significant potential benefits of automatic clustering is to reveal to the user, heretofore unseen structures in the knowledge-base that either give additional insight or indicate problems in its organization. In the MVP-CA methodology both syntactic and semantic criteria are used for obtaining meaningful partitionings. A framework is provided through the concept focus list, wherein the developer can identify meaningful semantic concepts within a particular knowledge base, which can then form the basis for further clustering. The concept-focus list thus allows a user now to control the patterns that go into defining the distance between rules. This provides for more control in the eventual clustering that takes place. Now one can either sharpen a current viewpoint (by identifying highly dispersed "noise" patterns and weeding them out of the clustering process) or filter primary concepts in the rulebase to allow secondary concepts to emerge.

Our methodology for MVP-CA can be summarized as follows. First, form the best cluster possible (using statistical measures of cohesiveness and coupling) for a particular distance metric. If "almost" perfect groupings are obtained, identify the "noise" patterns that are interfering with the formation of perfect groups. This is done by examining the dispersion statistics generated from the clustering. Next, with the "best" clustering(s), examine the dispersion values of all patterns. Identify primary viewpoint for this clustering. Remove the concept patterns that are responsible for the pri-

mary viewpoint from all the rules. Now repeat the clustering to find the secondary (tertiary, etc.) viewpoints. Run the knowledge base with other distance metrics applying the above procedure to get further viewpoints on the rulebase.

4 Focus of this Project

Currently both these systems, IMRD-ES and CFAR-ES, are used independently for target detection and radar control. It is the objective of Rome Labs to integrate the two systems on more current software/hardware platforms in such a way that:

- Real-time constraints for radar control are observed through possibly parallelizing much of the signal processing code.
- Artificial intelligence techniques from the two systems are merged and enhanced to make the system more adaptive for real world radar applications.
- Sound software engineering techniques are used in the integration process so that the system is easily maintainable and extensible.

MVP-CA tool was chosen to be used to identify the common functionalities in IMRD-ES and CFAR-ES code by partitioning these codes into semantically meaningful clusters. The multi-viewpoint approach, as discussed in the previous section, utilizes clustering analysis techniques to group rules which share significant common properties and then it identifies the concepts

which underlie these groups. Exposing underlying domain and subdomain information of the systems in this manner increases the odds of detecting common regions of functionalities in the two codes. This leads to a better reuse of existing code.

5 Application of MVP-CA to IMRD-ES and CFAR-ES

5.1 Results of using MVP-CA on IMRD-ES

Under this contract, Pragati, Inc. was successful in analyzing the structure of IMRD-ES using the MVP-CA tool. In order to run the MVP-CA tool on IMRD-ES, Prolog's BNF-grammar had to be acquired over the internet from Quintus Corporation. It took us about one month to build the frond-end interface to Pragati's clustering tool which could convert the Prolog code into the internal format for the tool. We also had to optimize the MVP-CA tool to enable large systems such as IMRD-ES to be analyzed faster and more efficiently.

After the front-end was built, we combined all the rules from different Prolog files into one file. This was done to enable MVP-CA tool to come up with its own clusters that are purely domain knowledge dependent and did not incorporate the control aspects of the program. Our experience has shown that generally developers of knowledge-bases, tend to partition the knowledge-base according to the run-time or phase aspects of the system. It is important, however, to expose the static viewpoint of the knowledge-base

since a lot of functional commonalities across different files get discovered in this manner.

The predicate clauses in IMRD-ES formed a very convenient basis for labeling the rules so that the clusters could be analyzed for their semantic content. In order to label forward chaining systems, we have had to abbreviate relevant domain knowledge in each rule. Only then a correct interpretation of the semantics of the the clusters could be made. In a backward chaining system such as Prolog, the goal predicate suggests very succintly the overall objective of the rule. Therefore we labeled all 337 rules with their left-hand side predicates.

Next, we ran IMRD-ES with different distance metrics in the MVP-CA tool so as to find the most suitable one. The distance metric that resulted in the most meaningful clusters was the consequent metric. This is because Prolog is a backward chaining system with all the predicates or goals to be fulfilled present on the left-hand side. Hence all domain knowledge in such a system can be expected to reside on the right- hand side where how the goal is being fulfilled is being elaborated.

Even though the consequent metric gave us some useful high-level clusters, it was not possible to get deeper into the code as most of the clauses on the right-hand-side were procedure calls to Pascal. All the control structure was found to reside in Prolog rules and the real computations were carried out in Pascal procedures.

The rules in Prolog were structured approximately into the following clusters as discovered by the MVP-CA tool (also shown in Figure 1):

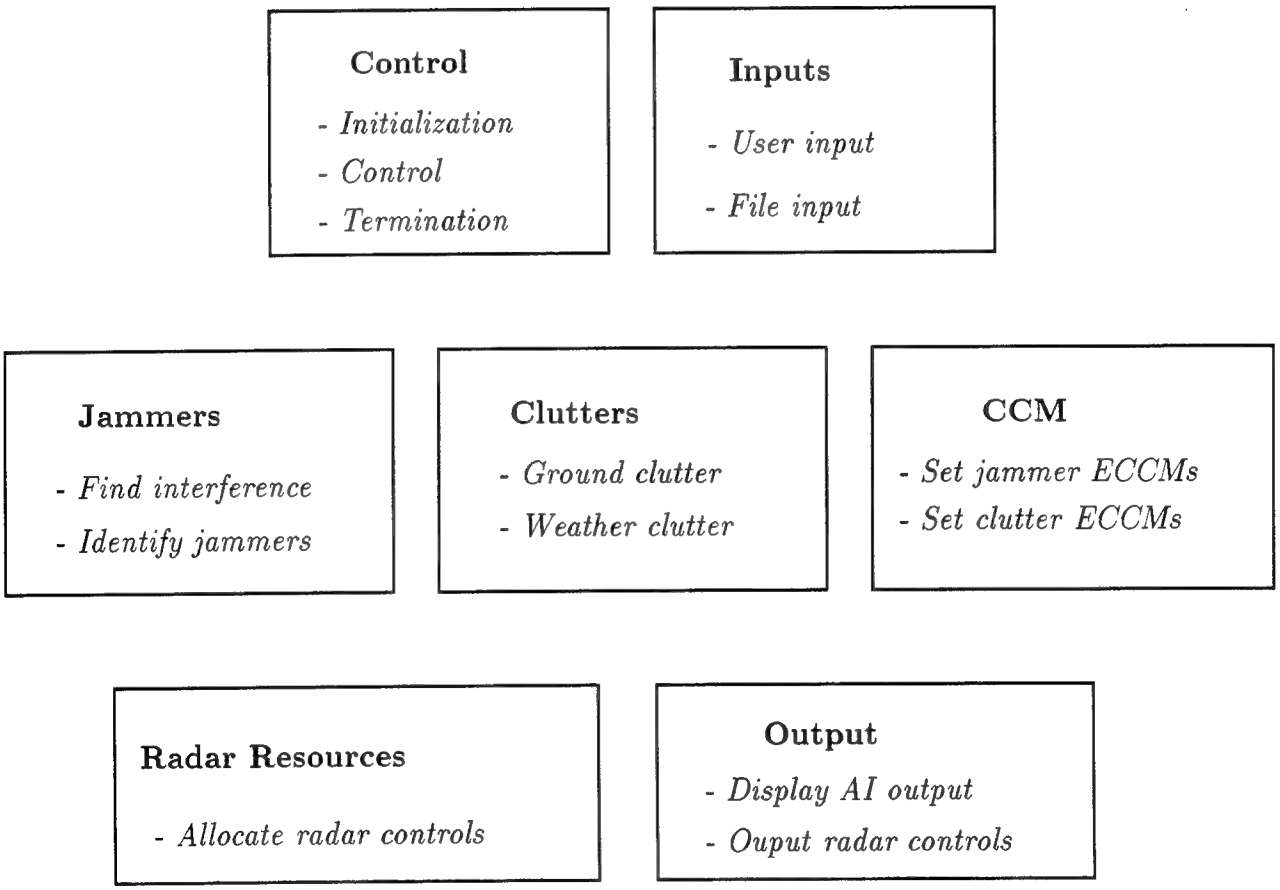

Figure 1: A viewpoint of the IMRD-ES Rulebase

- *Control rules* for performing initialization, control and termination of the program.
- *Input rules* for reading file input as well as user input.
- *Jammer detection rules* to find interferences and identify the types of jammers present.
- *Clutter detection rules* to discriminate ground and weather clutter.
- *Counter-counter measure rules* to set jammer as well as clutter electronic counter counter measures (ECCM).
- *Radar resource allocation rules* to apportion the resources of the radar according to the prioritization of the threat regions, as well as overall quality options set for the different priorities.
- *Output rules* to display the radar control and AI outputs onto a screen.

In the original partitioning of IMRD-ES by the developers, radar resource allocation rules, input-output rules and control rules were scattered across different files. Our clustering brought these rules into a single perspective.

This clustering of the IMRD-ES rules allowed us to gain an understanding of the structure of the AI portion of IMRD-ES. However, MVP-CA tool currently works only on rulebases. Most of IMRD-ES domain knowledge is in the Pascal routines while the control knowledge is in the 337 Prolog predicates. Thus, MVP-CA could focus only on the control part. In order to perform more extensive analysis, mechanisms are needed which can include

the information about data accesses made by the Pascal routines offline and feed it into the multi-view point tool.

5.2 Results of using MVP-CA on CFAR-ES

The CFAR-ES system was developed using Gensym's G2, and could not be subjected to a similar study for the following reasons: G2 is a closed system storing all the rules in the internal format. It does not provide a mechanism for extraction of the rules to perform an offline study. This is the reason that the CFAR-ES tapes sent by Rome Laboratories could not be installed on our system. Since the CFAR-ES was built from within the Gensym shell, the first two tapes could not be read using the Unix environment. The ASCII version of the code had to be extracted through the "inspect" facility in Gensym. Finally, a readable version of the tape was received. However, there were only six rules to be installed and these could not be analyzed as they had been pulled out of their procedural environment. Hence they were out of context and analyzing them could give very misleading results.

It became evident to us during our trip to Rome in May 1994 to get a demonstration of the CFAR-ES that none of the CFAR rules written in G2 were in a stand alone form, i.e., they were embedded in C procedure calls. This design possibly precluded the use of any efficient pattern matching scheme provided by G2's inference engine.

We also tried to access CFAR-ES code through the internet connection and analyze it by running it on Rome Laboratories Sparc station. However, due to security reasons all such access had been suspended and we were

unable to run the software from off-site.

6 Results of the Analysis

6.1 Issues in Integration of IMRD-ES with CFAR-ES

As discussed in the earlier section, we were able to analyze the IMRD-ES code. However, since the CFAR-ES code could not be installed entirely, we had to base our understanding of CFAR-ES on documentations and papers provided to us by Rome Laboratories.

One of the major commonalities found in the two codes is that both use clutter (ground and weather based) information. In IMRD-ES the clutter information as generated by processing the CPI2 beam is used to set up the electronic counter-counter measures (ECCMs). Similarly, clutter information along with other geographical statistical information is used by CFAR-selection expert system to choose the appropriate CFAR algorithm to execute. Thus, in an integrated system, this information can be generated once for each beam and updated and used by both codes. An integrated system is depicted in Figure 2.

A possible timeline for the preprocessing of the beam scan data and the execution of the IMRD-ES and CFAR-ES is shown in Figure 3 (not drawn to scale). Assuming independent hardware for the two functions, as in the current system, the timeline shows how the preprocessing can be done in parallel with the execution of the integrated IMRD-ES and CFAR-ES codes.

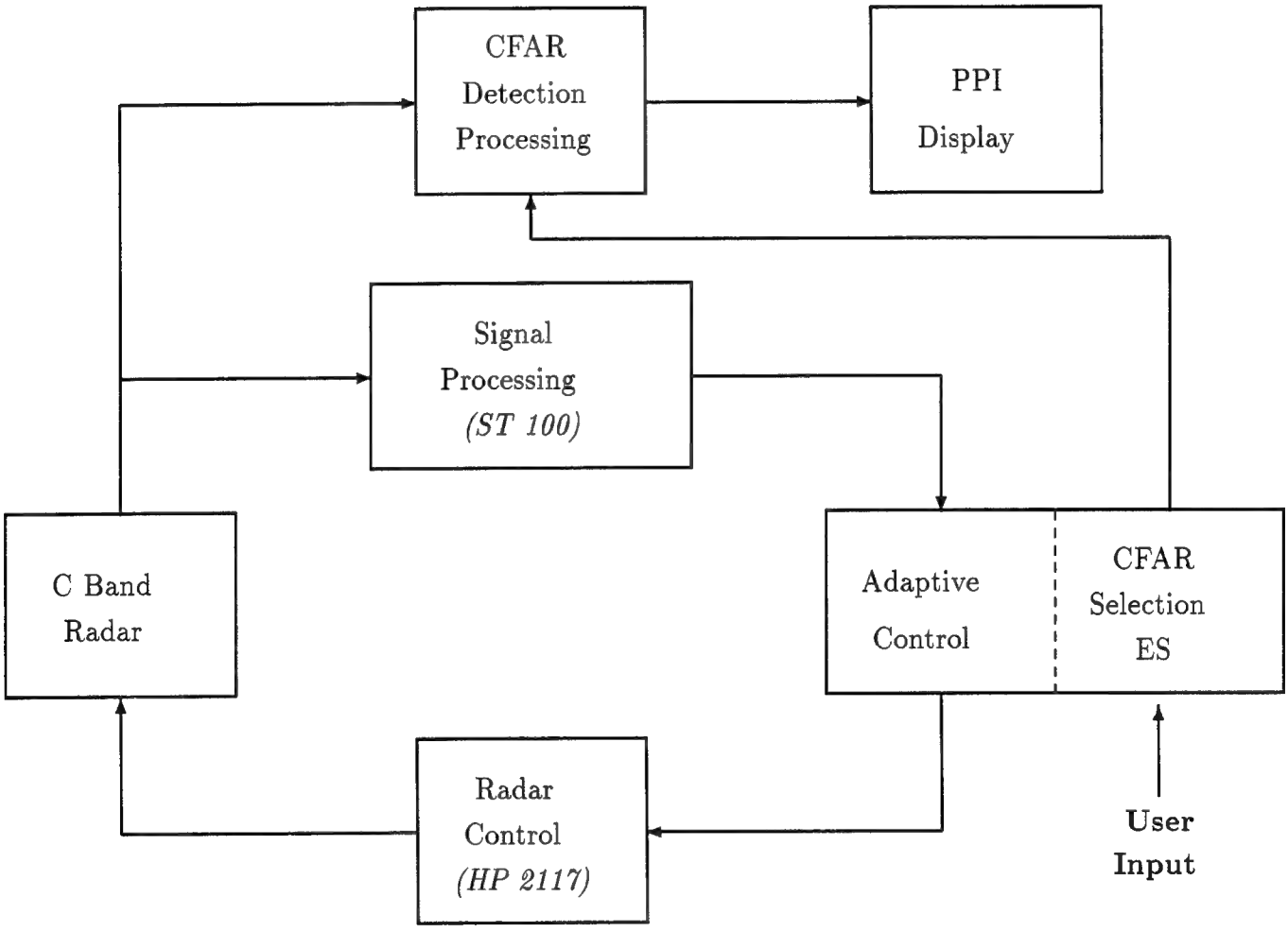

Figure 2: Suggested Integeration of IMRD-ES and CFAR-ES codes

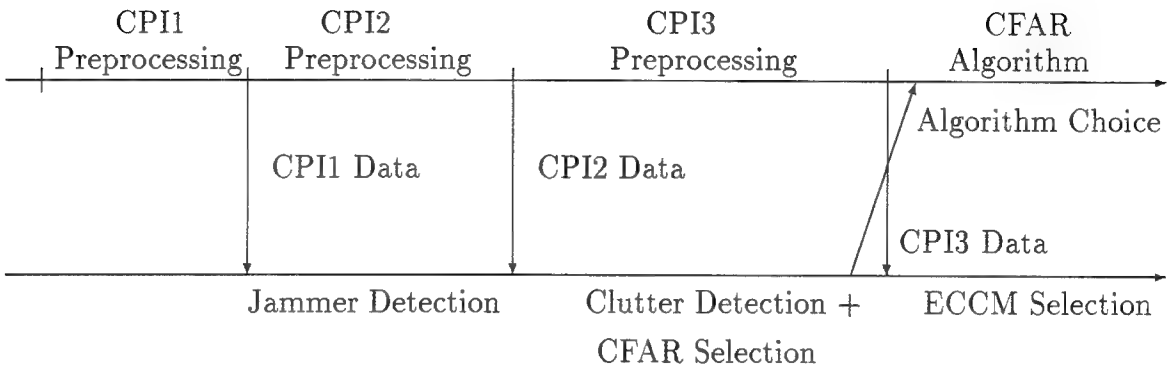

Figure 3: Timeline for preprocessing and IMRD-ES/CFAR-ES (not drawn to scale)

6.2 Disadvantages of Prolog

There is no structuring mechanism in Prolog which makes it very difficult to modify any software because interdependencies are not obvious upfront. Moreover, "cuts" and "fail" type of constructs limit the comprehensibility of code even further. Ideally, all such low-level control should be exercised through the inference engine. Furthermore, Prolog provides no direct support for real-time constraints which can be a very big drawback considering the time constraints that need to be satisfied in the new system. Also there is no direct support for fuzzy logic or uncertainty. Since the fidelity of radar data is not that high, due to distortions produced in them by environmental conditions, any conclusions that are drawn about possible threats, should be qualified with a degree of confidence level.

6.3 Disadvantages of G2

Gensym's G2 provides no real-time support at the level required by Rome Laboratories radar surveillance systems. Also there is no fuzzy logic incorporation. The biggest drawback that we see in G2 is that the system stores all information in an internal format making extraction of knowledge very difficult. Even though it is an object-oriented system, an external verification and validation of the knowledge-base is impossible in the current setup.

6.4 Expert System Characteristics Desired

Based on our knowledge of IMRD-ES and CFAR-ES we advocate that the following features will be desirable for the expert system shell choice:

- real-time constraints
- supports object-oriented design
- good integration with C or C++ routines
- incorporates fuzzy logic or at least uncertainty handling
- extensibility or modifiability of code
- maintainability and system support
- backward chaining system, since fidelity of data is not high enough to warrant a forward-chaining data-driven system.

7 Conclusions

In this project, we have used Pragati's MVP-CA tool to structure and analyze the IMRD-ES rulebase used for resource allocation in radar signal processing Based on our analysis of IMRD-ES software along with the understanding gained from the CFAR-ES documentation, we have suggested possible mechanisms for integrating the two systems. These include removal of redundant computation such as clutter detection which is common in both codes. We have also suggested potential situations where parallelism between the preprocessing and algorithm steps can be readily exploited if the integrated system is ported to the appropriate hardware.

Based on our analysis, we have also shown the shortcomings of the current expert system shells used in the two systems. We have suggested characteristics desired in an expert system shell which can be used to efficiently implement the integrated system such that it meets the desired realtime constraints.

References

[1] V. R. Basili and B. T. Perricone. Software Errors and Complexity: An Empirical Investigation. *Communications of the ACM*, 1(27):42–52, January 1984.

[2] B. Chandrasekharan. Generic tasks in knowledge-based reasoning: High-level building blocks for expert systems design. *IEEE Expert*, Fall 1986.

[3] *Expert System Verfication Validation and Evaluation Handbook: Version 2 (Draft)*, June 1994.

[4] N. Leveson. Software Safety: What, Why and How. *Computing Surveys*, 2(18):125–164, June 1986.

[5] M. Mehrotra. Rule Groupings: A Software Engineering Approach towards Verification of Expert Systems. Technical Report NASA CR-4372, NASA Langley Research Center, Hampton, VA., May 1991.

[6] M. Mehrotra. Application of Multi-Viepoint Clustering Analysis to a Highway Maintenance System. Technical report, Pragati Final Report - to appear as an NTIS report also, Yorktown, VA., November 1994.

[7] M. Mehrotra and S. C. Johnson. Importance of Rule Groupings in Verification of Expert Systems. In *Notes for the AAAI-90 Workshop on Verification, Validation and Testing of Knowledge-Based Systems*, July 1990.

[8] M. Mehrotra and S. C. Johnson. Rule Groupings in Expert Systems. In *Proceedings, First CLIPS Users Group Conference*, Aug 1990.

[9] M. Mehrotra and C. Wild. Multi-view point clustering analysis. In *Proceedings, 1993 Goddard Conference on Space Applications of Artificial Intelligence*, pages 217–231, May 1993.

[10] M. Mehrotra and C. Wild. Multi-viewpoint clustering analysis. In *Notes for the AAAI-93 Workshop on Verification, Validation and Testing of Knowledge-Based Systems*, July 1993.

[11] M. Mehrotra and C. Wild. Analyzing knowledge-based systems using multi-viewpoint clustering analysis. *Journal of Systems and Software*, to appear in March 1995.

[12] M. Mehrotra, C. Wild, and D. Rosca. Role of clustering analysis in the verification of expert systems. In *Notes for the AAAI-92 Workshop on Verification, Validation and Testing of Knowledge-Based Systems*, July 1992.

[13] C. Wild, J. Chen, and D. Eckhardt. Reasoning about Software Specifications: A Case Study. *Proceedings of AIAA Computers in Aerospace VII Conference*, pages 297–306, October 1989.

*U.S. GOVERNMENT PRINTING OFFICE: 1995-610-126-50189

Rome Laboratory

Customer Satisfaction Survey

RL-TR-_____

Please complete this survey, and mail to RL/IMPS, 26 Electronic Pky, Griffiss AFB NY 13441-4514. Your assessment and feedback regarding this technical report will allow Rome Laboratory to have a vehicle to continuously improve our methods of research, publication, and customer satisfaction. Your assistance is greatly appreciated.
Thank You

_____

_____

Organization Name:_____(Optional)

Organization POC: _____(Optional)

Address:_____

1. On a scale of 1 to 5 how would you rate the technology developed under this research?

5-Extremely Useful 1-Not Useful/Wasteful

Rating_____

Please use the space below to comment on your rating. Please suggest improvements. Use the back of this sheet if necessary.

2. Do any specific areas of the report stand out as exceptional?

Yes___ No_____

If yes, please identify the area(s), and comment on what aspects make them "stand out."

3. Do any specific areas of the report stand out as inferior?

Yes___ No___

If yes, please identify the area(s), and comment on what aspects make them "stand out."

4. Please utilize the space below to comment on any other aspects of the report. Comments on both technical content and reporting format are desired.

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.